Récupération du contenu d'un site web

Il existe déjà de nombreux outils prêts à l'emploi pour extraire le contenu des sites web. Cependant, en les utilisant, nous ne pouvons pas mieux comprendre le processus sous-jacent. Si nous rencontrons des sites web complexes ou particuliers dans notre travail, leur utilisation pourrait ne pas donner les résultats souhaités. Nous avons besoin de réinventer la roue, pour mieux les apprendre et mieux les utiliser.

Jetons également un coup d'œil à certains outils existants.

Data Miner

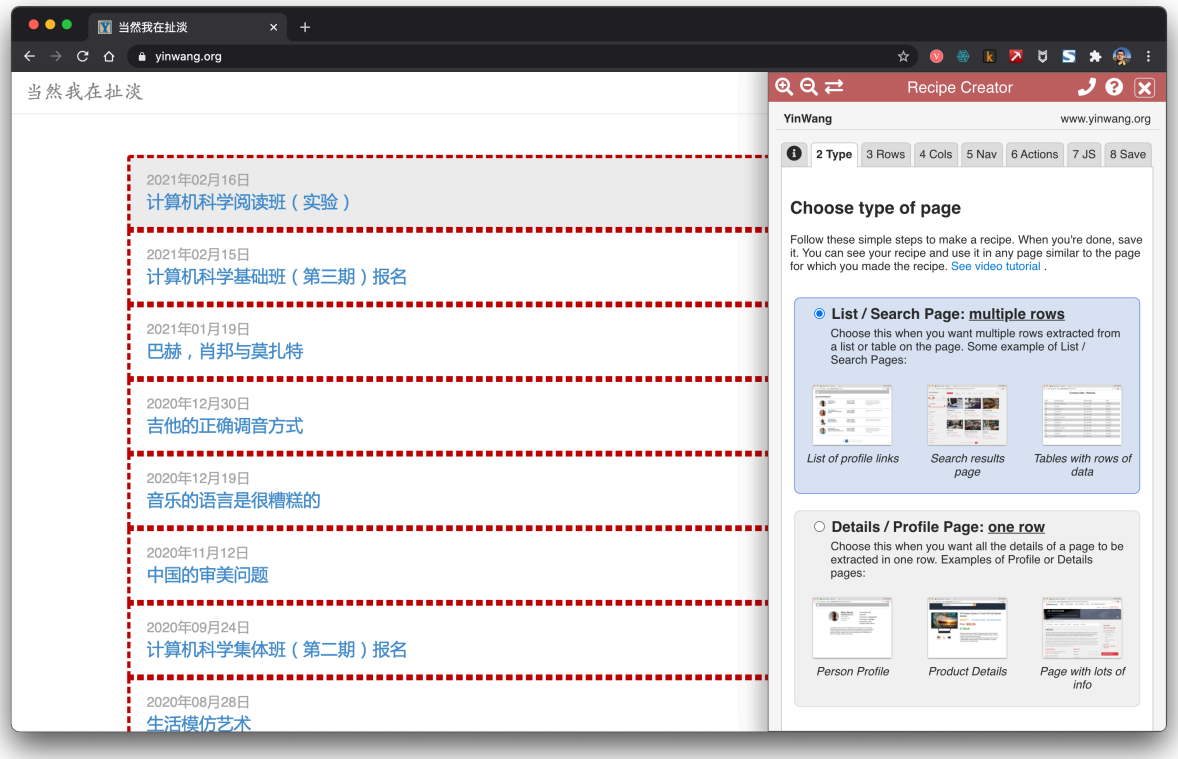

Figure 1: miner

`Data Miner` est une extension très pratique pour Chrome. Elle permet de récupérer facilement des liens et du contenu.

getbook

`getbook` est un outil très pratique pour créer des livres électroniques.

```
pip install getbook
```

```
{
  "title": "Mon Livre",
  "author": "John Doe",
  "description": "Ceci est un exemple de fichier book.json pour un projet de livre.",
  "language": "fr",
  "plugins": [
    "theme-default",
    "highlight",
    "search",
    "lunr"
  ]
}
```

```
{
  "uid": "book",
  "title": "Hello World",
  "author": "Armin",
  "chapters": [
    "http://lucumr.pocoo.org/2018/7/13/python/",
    "http://lucumr.pocoo.org/2017/6/5/diversity-in-technology",
  ]
}
```

```
getbook -f ./book.json --mobi
```

Ainsi, il est devenu pratique de transformer certains liens en livres électroniques. En utilisant `Data Miner` et `getbook`, l'un pour extraire les liens et l'autre pour les convertir en livres électroniques, il est désormais très facile de créer des livres électroniques.

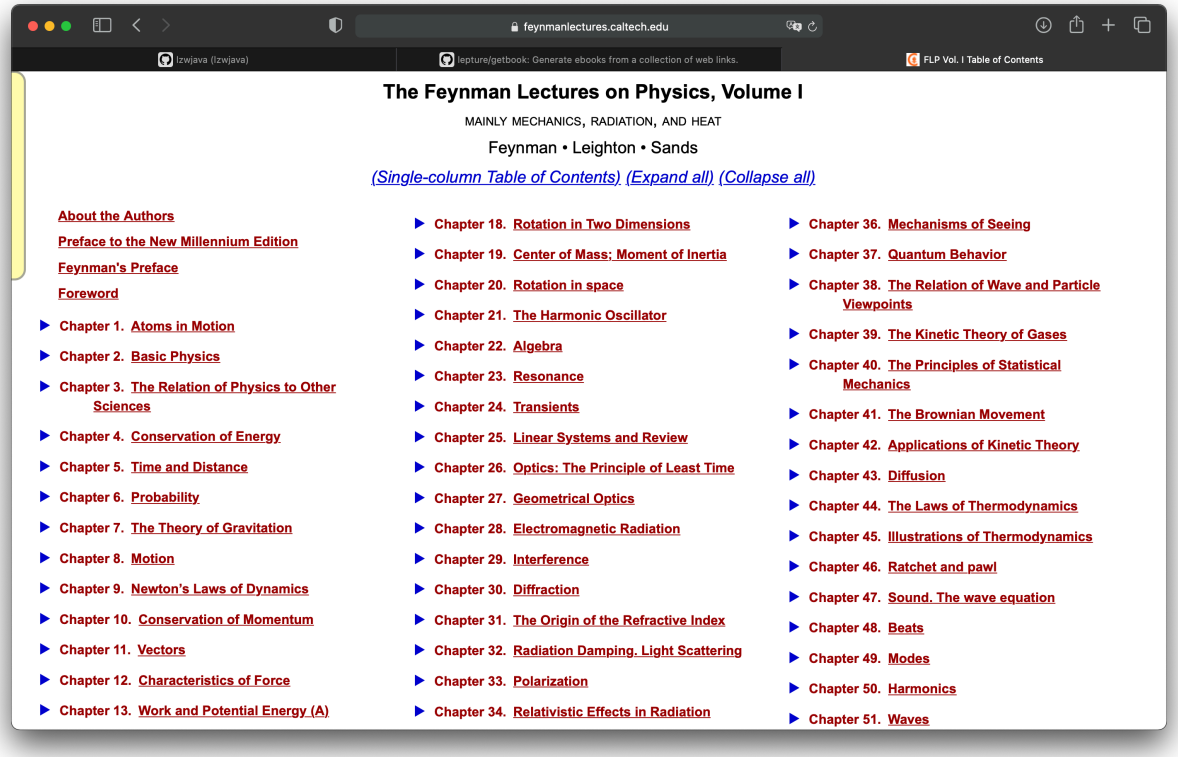
feynmanlectures.caltech.edu
lzwjava (lzwjava)
lepture/getbook: Generate ebooks from a collection of web links.
FLP Vol. I Table of Contents
The Feynman Lectures on Physics, Volume I
MAINLY MECHANICS, RADIATION, AND HEAT
Feynman • Leighton • Sands
(Single-column Table of Contents) (Expand all) (Collapse all)
About the Authors
Preface to the New Millennium Edition
Feynman's Preface
Foreword
Chapter 1. Atoms in Motion
Chapter 2. Basic Physics
Chapter 3. The Relation of Physics to Other Sciences
Chapter 4. Conservation of Energy
Chapter 5. Time and Distance
Chapter 6. Probability
Chapter 7. The Theory of Gravitation
Chapter 8. Motion
Chapter 9. Newton's Laws of Dynamics
Chapter 10. Conservation of Momentum
Chapter 11. Vectors
Chapter 12. Characteristics of Force
Chapter 13. Work and Potential Energy (A)
Chapter 18. Rotation in Two Dimensions
Chapter 19. Center of Mass; Moment of Inertia
Chapter 20. Rotation in space
Chapter 21. The Harmonic Oscillator
Chapter 22. Algebra
Chapter 23. Resonance
Chapter 24. Transients
Chapter 25. Linear Systems and Review
Chapter 26. Optics: The Principle of Least Time
Chapter 27. Geometrical Optics
Chapter 28. Electromagnetic Radiation
Chapter 29. Interference
Chapter 30. Diffraction
Chapter 31. The Origin of the Refractive Index
Chapter 32. Radiation Damping. Light Scattering
Chapter 33. Polarization
Chapter 34. Relativistic Effects in Radiation
Chapter 36. Mechanisms of Seeing
Chapter 37. Quantum Behavior
Chapter 38. The Relation of Wave and Particle Viewpoints
Chapter 39. The Kinetic Theory of Gases
Chapter 40. The Principles of Statistical Mechanics
Chapter 41. The Brownian Movement
Chapter 42. Applications of Kinetic Theory
Chapter 43. Diffusion
Chapter 44. The Laws of Thermodynamics
Chapter 45. Illustrations of Thermodynamics
Chapter 46. Ratchet and pawl
Chapter 47. Sound. The wave equation
Chapter 48. Beats
Chapter 49. Modes
Chapter 50. Harmonics
Chapter 51. Waves

Figure 2: fl

Les Cours de Physique de Feynman

Dans le chapitre « Projet pratique : Transformer les pages web des cours de physique de Feynman en livre électronique », nous avons appris comment convertir une page web rendue avec `mathjax` en un livre électronique. Nous poursuivons ici ce projet pour voir comment récupérer toutes les pages. Les cours de physique de Feynman sont répartis en trois volumes. L'image ci-dessus montre le sommaire du premier volume.

> http.client — Client du protocole HTTP
>
> Code source : Lib/http/client.py
>
> Ce module définit des classes qui implémentent le côté client des protocoles HTTP et HTTPS. Il n'est généralement pas utilisé directement — le module urllib.request l'utilise pour gérer les URL qui utilisent HTTP et HTTPS.
>
> Voir aussi : Le package Requests est recommandé pour une interface de client HTTP de plus haut niveau.

On peut voir que `requests` est une interface de plus haut niveau.

```
import requests

def main():
    r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
    print(r.status_code)

main()

```shell
401

import requests

def main():
    r = requests.get('https://github.com')
    print(r.status_code)
    print(r.text)

main()
```
```

```html
200
<html>
  ...
</html>
```

J'ai essayé, cela montre que l'interface de `requests` est fonctionnelle.

```
     <div class="toc-chapter" id="C03">
        <span class="triangle">
         â ¶
        </span>
        <a class="chapterlink" href="javascript:Goto(1,3)">
         <span class="tag">
          Chapitre 3.
         </span>
         La relation de la physique avec les autres sciences
        </a>
        <div class="sections">
         <a href="javascript:Goto(1,3,1)">
          <span class="tag">
           3-1
          </span>
          Introduction
         </a>
         <a href="javascript:Goto(1,3,2)">
          <span class="tag">
           3-2
          </span>
          Chimie
         </a>
         <a href="javascript:Goto(1,3,3)">
          <span class="tag">
           3-3
          </span>
          Biologie
         </a>
```

```
<a href="javascript:Goto(1,3,4)">
 <span class="tag">
  3-4
 </span>
 Astronomie
</a>
<a href="javascript:Goto(1,3,5)">
 <span class="tag">
  3-5
 </span>
 Géologie
</a>
<a href="javascript:Goto(1,3,6)">
 <span class="tag">
  3-6
 </span>
 Psychologie
</a>
<a href="javascript:Goto(1,3,7)">
 <span class="tag">
  3-7
 </span>
 Comment en est-on arrivé là ?
</a>
</div>
</div>
```

Voici le code HTML de la troisième section dans la page de sommaire. L'objectif est d'extraire les liens de chaque section à partir de ce code. `<a href="javascript:Goto(1,3,7)">`, on peut voir qu'il s'agit d'un lien hypertexte JavaScript.

```
https://www.feynmanlectures.caltech.edu/I_03.html
```

Ensuite, j'ai remarqué que le chemin de chaque chapitre suivait une logique bien définie. Par exemple, `I_03.html` représente le chapitre 3 du premier volume.

```
import requests
```

```
from bs4 import BeautifulSoup
from multiprocessing import Process

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1))
        p.start()
        p.join()

main()
```

Continuons avec l'écriture du code de scraping. Ici, nous utilisons `Process`.

```
    raise RuntimeError('''
RuntimeError:
        Une tentative a été faite pour démarrer un nouveau processus avant que
        le processus actuel n'ait terminé sa phase d'initialisation.

    Cela signifie probablement que vous n'utilisez pas fork pour démarrer vos
    processus enfants et que vous avez oublié d'utiliser l'idiome approprié
    dans le module principal :

        if __name__ == '__main__':
```

```
            freeze_support()
            ...

    La ligne "freeze_support()" peut être omise si le programme
    ne va pas être figé pour produire un exécutable.
```

```python
def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
    p.join()

if __name__ == "__main__":
    main()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Temps : ', stop - start)

if __name__ == "__main__":
    main()
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_04.html
...
scraping https://www.feynmanlectures.caltech.edu/I_51.html
scraping https://www.feynmanlectures.caltech.edu/I_52.html
Temps :  9.144841699
```

```
<div class="figure" id="Ch1-F1">
        <img src="img/FLP_I/f01-01/f01-01_tc_big.svgz">
```

the most information in the fewest words? I believe it is the *atomic hypothesis* (or the atomic *fact* , or whatever you wish to call it) that *all things are made of atomsâlittle particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another* . In that one sentence, you will see, there is an *enormous* amount of information about the world, if just a little imagination and thinking are applied.

Figure 1â1

To illustrate the power of the atomic idea, suppose that we have a drop of water a quarter of an inch on the side. If we look at it very closely we see nothing but waterâsmooth, continuous water. Even if we magnify it with the best optical microscope availableâroughly two thousand timesâthen the water drop will be roughly forty feet across, about as big as a large room, and if we looked rather closely, we would *still* see relatively smooth waterâbut here and there small football-shaped things swimming back and forth. Very interesting. These are paramecia. You may stop at this

Figure 3: fig

```
        <div class="caption empty">
         <span class="tag">
          Figure 1-1
         </span>
        </div>
</div>
```

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
```

```
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Temps : ', stop - start)

if __name__ == "__main__":
    main()
```

Regardez le lien.

```
    imgs = soup.find_all('img')
    for img in imgs:
        print(img)
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
<img id="TwitLink" src=""/>
<img id="FBLink" src=""/>
<img id="MailLink" src=""/>
<img id="MobileLink" src=""/>
<img id="DarkModeLink" src=""/>
<img id="DesktopLink" src=""/>
<img src="img/camera.svg"/>
<img src="img/FLP_I/f01-00/f01-00.jpg"/>
<img data-src="img/FLP_I/f01-01/f01-01_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-02/f01-02_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-03/f01-03_tc_big.svgz"/>
```

```
<img data-src="img/FLP_I/f01-04/f01-04_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-05/f01-05_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-06/f01-06_tc_big.svgz"/>
<img class="first" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_a.svgz"/>
<img class="last" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_b.svgz"/>
<img data-src="img/FLP_I/f01-08/f01-08_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-09/f01-09_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-10/f01-10_tc_big.svgz"/>
```

https://www.feynmanlectures.caltech.edu/img/FLP_I/f01-01/f01-01_tc_big.svgz

```
Interdit
```

Vous n’avez pas l’autorisation d’accéder à cette ressource.

Apache/2.4.38 (Debian) Server à www.feynmanlectures.caltech.edu Port 443

```shell
% pip install selenium
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/site-packages (from selenium) (1.24.1
Installing collected packages: selenium
Successfully installed selenium-3.141.0

export CHROME_DRIVER_HOME=$HOME/dev-env/chromedriver
export PATH="${PATH}:${CHROME_DRIVER_HOME}"

% chromedriver -h
Usage : chromedriver [OPTIONS]
```

Options –port=PORT port sur lequel écouter –adb-port=PORT port du serveur adb –log-path=FILE écrire les logs du serveur dans un fichier au lieu de stderr, augmente le niveau de log à INFO –log-level=LEVEL définir le niveau de log : ALL, DEBUG, INFO, WARNING, SEVERE, OFF –verbose logger de manière verbeuse (équivalent à –log-level=ALL) –silent ne rien logger (équivalent à –log-level=OFF) –append-log ajouter au fichier de log au lieu de le réécrire –replayable (expérimental) logger de manière verbeuse et ne pas tronquer les chaînes longues pour que le log puisse être rejoué. –version afficher le numéro de version et quitter –url-base

préfixe de base pour les URL des commandes, par exemple wd/url –readable-timestamp ajouter des horodatages lisibles au log –enable-chrome-logs afficher les logs du navigateur (remplace les autres options de log) –allowed-ips liste d'adresses IP distantes autorisées à se connecter à ChromeDriver, séparées par des virgules

```python
# Le code reste en anglais, comme dans l'original

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located

with webdriver.Chrome() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get("https://google.com/ncr")
    driver.find_element(By.NAME, "q").send_keys("cheese" + Keys.RETURN)
    first_result = wait.until(presence_of_element_located((By.CSS_SELECTOR, "h3>div")))
    print(first_result.get_attribute("textContent"))

# Le code reste en anglais, comme dans l'original.

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
import urllib

def main():
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)
    driver.get("https://www.feynmanlectures.caltech.edu/I_01.html")
    elements = driver.find_elements(By.TAG_NAME, "img")
    # print(dir(elements[0]))
    print(driver.page_source)
```

```
    i = 0
    for element in elements:
        # src = element.get_attribute('src')
        element.screenshot(f'images/{i}.png')
        i +=1
    driver.close()
main()

from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit
from pathlib import Path
from selenium import webdriver
from selenium.webdriver.common.by import By

def img_path(chapter):
    return f'./chapters/{chapter}/img'

def img_name(url):
    splits = url.split('/')
    last = splits[len(splits) - 1]
    parts = last.split('.')
    name = parts[0]
    return name

def download_images(driver: webdriver.Chrome, chapter):
    path = img_path(chapter)
    Path(path).mkdir(parents=True, exist_ok=True)

    elements = driver.find_elements(By.TAG_NAME, "img")
    for element in elements:
        src = element.get_attribute('src')
        name = img_name(src)
        element.screenshot(f'{path}/{name}.png')
```

USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, comme Gecko) Version/14.0.3 Safari/605.1.15'

```
def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    driver = webdriver.Chrome()
    driver.get(url)
    page_source = driver.page_source
    Path(f'./chapters/{chapter_str}').mkdir(parents=True, exist_ok=True)
    print(f'scraping {url}')

    download_images(driver, chapter_str)

    soup = BeautifulSoup(page_source, features='lxml')
    imgs = soup.find_all('img')
    for img in imgs:
        if 'src' in img.attrs or 'data-src' in img.attrs:
            src = ''
            if 'src' in img.attrs:
                src = img.attrs['src']
            elif 'data-src' in img.attrs:
                src = img.attrs['data-src']
                del img.attrs['data-src']
            name = img_name(src)
            img.attrs['src'] = f'img/{name}.png'

    f = open(f'./chapters/{chapter_str}/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

    driver.close()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(2)]
    for p in ps:
        p.start()
```

```
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Temps : ', stop - start)

if __name__ == "__main__":
    main()
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_02.html
Temps : 21.478510914999998
```

```
 errpipe_read, errpipe_write = os.pipe()
OSError: [Errno 24] Trop de fichiers ouverts
```

```
 % ulimit a
ulimit: nombre invalide : a
lzw@lzwjava feynman-lectures-mobi % ulimit -a
-t: temps CPU (secondes)              illimité
-f: taille de fichier (blocs)         illimité
-d: taille du segment de données (kbytes) illimité
-s: taille de la pile (kbytes)        8192
-c: taille du fichier core (blocs)    0
-v: espace d'adressage (kbytes)       illimité
-l: taille mémoire verrouillée (kbytes) illimité
-u: processus                         2784
-n: descripteurs de fichiers          256
```

```
12
download_images
12
mathjax2svg
latexs 128
make_svg 0
insert_svg 0
make_svg 1
insert_svg 1
```

```
make_svg 2
insert_svg 2
make_svg 3
insert_svg 3
convert

12
download_images
12
mathjax2svg
latexs 0
latexs 0
convert
Temps : 11.369145162

% grep --include=\*.html -r '\$' *
43/I_43.html:une longue période de temps $T$, ont un certain nombre, $N$, de collisions. Si nous
43/I_43.html:le nombre de collisions est proportionnel au temps $T$. Nous aimerions
43/I_43.html:Nous avons écrit la constante de proportionnalité comme $1/\tau$, où
43/I_43.html:$\tau$ aura les dimensions d'un temps. La constante $\tau$ est le
43/I_43.html:il y a $60$ collisions ; alors $\tau$ est une minute. Nous dirions
43/I_43.html:que $\tau$ (une minute) est le

□□  E21018: Échec de la création d'un domaine Mobi amélioré lors de l'analyse du contenu du fichier. C
Avertissement   W28001: Le lecteur Kindle ne prend pas en charge le style CSS spécifié dans le contenu.
Avertissement   W29004: Balise ouverte fermée de force : <span amzn-src-id="985"> Fichier : /private/var
Avertissement   W29004: Balise ouverte fermée de force : <p amzn-src-id="975"> Fichier : /private/var/f

Avertissement   W14001: Problème de lien hypertexte non résolu :  /private/var/folders/_3/n3b7dq8x6652d
Avertissement   W14001: Problème de lien hypertexte non résolu :  /private/var/folders/_3/n3b7dq8x6652d
Avertissement   W14001: Problème de lien hypertexte non résolu :  /private/var/folders/_3/n3b7dq8x6652d

<span class="disabled" href="#Ch1-F1">
          1-1
</span>

Rasterisation de 'OEBPS/84b8b4179175f097be1180a10089107be75d7d85.svg' en 1264x1011
```

```
Rasterisation de 'OEBPS/23a4df37f269c8ed43f54753eb838b29cff538a1.svg' en 1264x259
Traceback (dernier appel le plus récent) :
  Fichier "runpy.py", ligne 194, dans _run_module_as_main
  Fichier "runpy.py", ligne 87, dans _run_code
  Fichier "site.py", ligne 39, dans <module>
  Fichier "site.py", ligne 35, dans main
  Fichier "calibre/utils/ipc/worker.py", ligne 216, dans main
  Fichier "calibre/gui2/convert/gui_conversion.py", ligne 41, dans gui_convert_override
  Fichier "calibre/gui2/convert/gui_conversion.py", ligne 28, dans gui_convert
  Fichier "calibre/ebooks/conversion/plumber.py", ligne 1274, dans run
  Fichier "calibre/ebooks/conversion/plugins/mobi_output.py", ligne 214, dans convert
  Fichier "calibre/ebooks/conversion/plugins/mobi_output.py", ligne 237, dans write_mobi
  Fichier "calibre/ebooks/oeb/transforms/rasterize.py", ligne 55, dans __call__
  Fichier "calibre/ebooks/oeb/transforms/rasterize.py", ligne 142, dans rasterize_spine
  Fichier "calibre/ebooks/oeb/transforms/rasterize.py", ligne 152, dans rasterize_item
  Fichier "calibre/ebooks/oeb/transforms/rasterize.py", ligne 185, dans rasterize_external
  Fichier "calibre/ebooks/oeb/base.py", ligne 1092, dans bytes_representation
  Fichier "calibre/ebooks/oeb/base.py", ligne 432, dans serialize
TypeError: impossible de convertir un objet 'NoneType' en bytes

% kindlepreviewer feynman-lectures-on-physics-volumn-1.epub -convert
Vérification des arguments spécifiés.
Pré-traitement en cours.
Traitement de 1/1 livre(s).
Livre converti avec des avertissements ! : /Users/lzw/projects/feynman-lectures-mobi/feynman-lectures-o
Post-traitement en cours.
Écriture des fichiers de sortie/logs dans /Users/lzw/projects/feynman-lectures-mobi/output

Nettoyage du manifeste...
Suppression des fichiers inutilisés du manifeste...
Création de la sortie AZW3...
Sérialisation des ressources...
Découpage du balisage sur les sauts de page et les limites de flux, si nécessaire...
Création de la sortie KF8
    Génération du balisage KF8...
La table de balises n'a pas d'aide et une taille de bloc trop grande. Ajout malgré tout.
La table de balises n'a pas d'aide et une taille de bloc trop grande. Ajout malgré tout.
```

```
La table de balises n'a pas d'aide et une taille de bloc trop grande. Ajout malgré tout.
    Compression du balisage...
    Création des index...
```